جامعة دمشق

كلية الهندسة المعلوماتية

قسم النظم و الشبكات الحاسوبية

السنة الخامسة



Real Time System

Barber Shop

إعداد :

مصطفى محمد نجم

Moustafa-MN@hotmail.com

2008-2009

• مقدمة :

يعتبر التزامن من أهم المشاكل التي تواجه المبرمج أثناء تنفيذ البرامج المختلفة، و بخاصة التي تشترك فيها أكثر من مهمة بموارد محددة.

هناك لغات برمجة صممت لتطبيقات نظم الزمن الحقيقي ،حيث أن تحوي اللغة في صلبها منطق المزامنة و قيود على الوصول إلى الموارد،و في هذه اللغات يقتصر دور المبرمج على توصيف العمل المطلوب و يترك مهمة المزامنة للغة بما تملكه من إمكانيات ،و لكن على المبرمج توصيف العمل بشكل دقيق .

من هذه اللغات لغة ADA التي سنستخدمها في برمجة حل لمسألة الحلاق BarberShop .

○ مقدمة و تاريخ مختصر عن ADA :

لغة Ada هي لغة عالية المستوى صممت من أجل برمجة الأنظمة ذات الزمن الحقيقي، و على مستوى عال و ضخم.

الاسم ADA مشتق من أوغستا ADA بيرن ابنة الأديب لورد بيرن، و عرفت بأول مبرمجة في العالم.

اخترعت ADA نتيجة تصور وكالة الدفاع الأمريكية أنه لا توجد لغة مناسبة جدا لتطبيقات أنظمة الزمن الحقيقي لنظم التحكم و الأنظمة Embedded Systems

Embedded Systems: التي هي عبارة عن نظام كمبيوتر موجود داخل نظام ما مثل الفرن الذكي أو الصواريخ الموجهة.



❖ مسألة الحلاق : Barber Shop Problem

توصيف المسألة :

نريد تمثيل عمل صالون الحلاق ،هذا الصالون يحوي عدد من الحلاقين N Barbers و كذلك عدد مساوي من كراسي الحلاقة N barber's chair ،و هناك غرفة انتظار تحوي عدد من الكراسي M waiting chairs.

- إذا لم يكن هناك زبائن ينام جميع الحلاقين،ريثما يأتي زبون.
- إذا كان جميع الحلاقين مشغولين ،و كان هناك كراسي شاغرة في غرفة الانتظار فيمكن للزبون أن يجلس على أحد الكراسي و ينتظر.
- إذا كان الحلاق نائم و جاء زبون ،فإن الزبون يقوم بإيقاظه.
- إذا انتهى أحد الحلاقين من الحلاقة لزبون فإنه يقوم بإيقاظ أحد الزبائن المنتظرين في غرفة الانتظار.و إذا لم يجد أي زبون فإنه يخلد للنوم،و يستغرق في الأحلام.
- إذا جاء زبون و وجد غرفة الانتظار ممتلئة و جميع الكراسي مشغولة ،فإن الزبون ينتظر لفترة على باب المحل T ثانية،خلال هذه الفترة إذا لم يفرغ أي كرسي فإنه يغادر المحل (الرجاء العودة في وقت لاحق..!!).

إذا من معطيات المسألة يمكن أن نلخص النقاط الأساسية :

- N barber , M customer .
- لدينا فترة انتظار للزبون T second .

ملاحظة :

يمكن أن نعتبر جميع الكراسي في المحل هي كراسي انتظار و لا نفرق بين كرسي انتظار و كرسي حلاق لأنها من حيث المنطق جميعها راسي موجودة داخل المحل.

و قد لاحظت ذلك من خلال الحل.

بنى المعطيات المستخدمة :

نستخدم في الحل مايلي :

- ## الوحدة المحمية Protected Unit :

– إن الوحدة المحمية تشبه إلى حد ما مفهوم التغليف في لغات البرمجة التقليدية .

– تحوي الموارد المشتركة (التي نحتاج إلى تطبيق التزامن عليها و منع الوصول إليها من قبل أكثر من مهمة بنفس الوقت) ،و التوابع و الإجرائيات التي تتعامل معها.

يمكن هنا أن نستخدم ثلاثة أنواع من التعريفات:

Function : للقراءة فقط ،لا يمكنه التعديل ،و هو يعيد قيمة.

Procedure: للكتابة ،يمكنه التعديل على المتحولات .

Entry : و هو عبارة عن procedure مع شرط (لا يتم الدخول إليها إلا عند تحقق الشرط).

كما نعرف المتحولات ضمن نطاق Private .

– نعرف الوحدة المحمية : **barbershop**

```
protected barbershop is
    entry take_chair;
    procedure leave;
    function howmany return integer;
    private
    num: integer:=0;
    chair_Max : integer:=5;
end barbershop;
```

- المتحولات : Private
  - **num** : عدد الزبائن في غرفة الانتظار ،و قيمته الابتدائية = 0 .
  - **chair_Max** :عدد الكراسي في غرفة الانتظار ،و هو عدد ثابت.
- التوابع و الإجرائيات:

- entry take_chair when (num<chair_Max)

و هو عبارة عن Entry يقوم بزيادة عدد المنتظرين ،يطلبه الزبون عندما يأتي إلى المحل ليحجز كرسي .

و لا يتم الدخول إليه إلا إذا كان عدد المنتظرين أقل من عدد الكراسي المتاحة (هناك كراسي شاغرة)

- function howmany return integer

و هو عبارة عن تابع يعيد عدد المنتظرين في غرفة الانتظار (يقابل Getter في لغات البرمجة التقليدية و الذي نستخدمه للحصول على قيمة متحول خاص private).

- procedure leave

و هي عبارة عن إجرائية يستدعيها الزبون عندما يغادر ،و تقوم بإنقاص عدد المنتظرين بمقدار 1 .

• المهمة Task :

- المهمة هي الوحدة التنفيذية في Ada و يبدأ تنفيذها من نقطة تعريفها (أو لحظة خلقها أي لا تحتاج إلى تشغيل من قبل المبرمج كما في النيسب Thread و الذي يتم تشغيله من خلال Start)
- بما أنه لدينا عدة زبائن و عدة حلاقين لذلك نستخدم Task Type و التي تعرف لنا نمط مهمة ثم نخلق مهام من هذا النمط.
- سنعرف نمط مهمة للحلاق و نمط مهمة للزبون.

- مهمة الحلاق : **barber**

```
task type barber(id:integer) is
    entry haircut;
end barber;
```

تحوي هذا المهمة Entry واحد و هو يمثل عملية الحلاقة حيث نعرف Accept له في جسم المهمة و داخلها نقوم بعملية انتظار فقط (تمثل فترة الحلاقة).

تأخذ هذه المهمة كدخل رقم مميز (فقط لتمييزها أثناء الطباعة).

- **مهمة الزبون : customer**

الزبون إما يدخل و يجلس على أحد الكراسي (في حال كان هناك كراسي شاغرة) و يطلب أحد الحلاقين (من خلال عملية select).

أو في حال كانت غرفة الانتظار ممتلئة : فإنه ينتظر لفترة زمنية ،و في حال لم يفرغ كرسي فإنه يغادر دون حلاقة.

- نستخدم هنا مفهوم **Rendezvous** للمزامنة بين الإجرائيات.

:Rendezvous

و هي طريقة للمزامنة بين مهمتين هما : المستدعي caller و المستدعى callee.

المستدعي هنا هو الزبون و المستدعى هو الحلاق.

حيث نعرف entry في المستدعى تمثل خدمة يقدمها للزبائن (و هي الحلاقة هنا haircut).

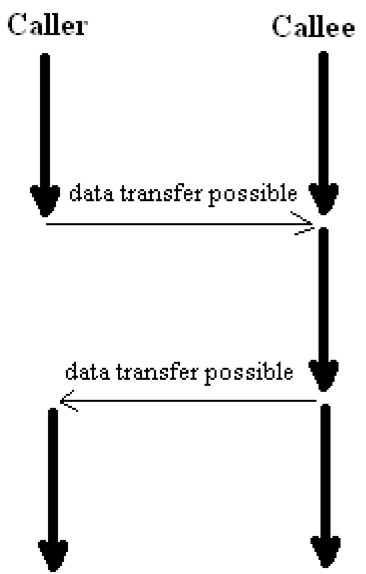
المستدعى ينتظر إرسال طلب من أحد الزبائن ليقوم بقبول الطلب و تخديمه (المستدعى ينتظر عند العبارة التي تمثل قبول الطلب و ينام ما لم يكن هناك طلبات).

المستدعي بدوره يرسل طلب و ينتظر حتى يتم تخديمه (أي يصف بالدور و ينام ثم يقوم المستدعى بإيقاظه عندما يحين دوره).

يمكن تمثيله بالشكل التالي :

أي يرسل المستدعي الطلب و ينتظر ريثما ينتهي المخدم من تخديمه(عندئذ يمكنه متابعة عمله).

هذا ضروري حتى لا نسمح للزبون بالذهاب قبل أن تتم عملية حلاقة شعره.



سيناريو العمل:

- **الزبون :**

– يأتي الزبون فيطلب take_chair (هنا نضمن أنه وحده من يعدل على num لأنها موجودة ضمن PU)

– إما يجد مكان فيدخل و يجلس و ينتظر أحد الحلاقين حتى يحلق له.

– أو لا يجد مكان فينتظر على الباب فترة زمنية علها تفرغ أحد الكراسي ضمن هذه الفترة .

▪ إذا فرغت أحد الكراسي خلال فترة الانتظار ،يدخل الزبون المنتظر ،و بالتالي لم يذهب الانتظار عالفاضي

▪ إذا انتهت فترة الانتظار و لم يفرغ أي كرسي ،يغادر الزبون المنتظر ،دون حلاقة ..!!

– عندما يدخل الزبون يقوم بإيقاظ الحلاق (في حال كان نائم) من خلال إرسال طلب.

– أو يقوم بالانتظار على أحد الحلاقين و ينام ،و الحلاق بدوره يوقظه من خلال عملية Accept.

- **الحلاق:**

– ينفذ إجرائية تخديم الطلبات باستمرار:

– فإذا لم يكن هناك طلبات (لا يوجد زبائن) فإنه ينام (يتوقف عند هذه النقطة حتى لحظة وصول طلب).

(عندما يأتي زبون فإنه يوقظ الحلاق من خلال إرسال طلب .. الحلاق يكون مسبقا منتظرا هذه اللحظة).

– أو يقوم بإيقاظ أحد الزبائن النائمين (المنتظرين) ليحلق له شعره.

– يقوم الحلاق بتخديم الطلب (حلاقة شعر الزبون).

– ثم يكرر العملية ليأخذ زبون آخر (في حال تواجده).

:خوارزمية الحل

Customer	Barber
Task type customer; Task body customer is select Barbershop.take_chair; select Barber1.haircut; Barbershop.leave; or Barber2.haircut; Barbershop.leave; or Barber3.haircut; Barbershop.leave; or delay T; end customer;	Task type barber is entry haircut; end barber ; Task type barber is loop If (barbershop.howmany=0) then put_line("sleeeeep"); end if; accept haircut do delay (haircut_time); end haircut; end loop; end barber;

✓ ملاحظة : يتم تحرير الكرسي بعد الحلاقة لأننا نعتبر أن كراسي الحلاقة هي من كراسي الانتظار و لا نفرق بينهما ،و بالتالي في حال كانت الغرفة ممتلئة لا يمكن دخول زبون جديد إلا بعد انتهاء أحد الزبائن من الحلاقة و مغادرته المحل و هذا منطقي. (ما يحدث في الواقع).

الحل باستخدام Rendezvous :

```
-- Barbershop program Using Ada (Rendezvous)
--Author Moustafa Najm ......... M.N : Moustafa-MN@hotmail.com
with Ada.Text_IO;
use Ada.Text_IO;

procedure Barber_Main is
  ---------------------------------- Protected Unit barbershop ----------------------------
  --barbershop declaration
  protected barbershop is
    entry take_chair;
    procedure leave;
    function howmany return integer;

    private
    num: integer:=0;
    chair_Max : integer:=5;
  end barbershop;

  --barbershop body
  protected body barbershop is

    entry take_chair when (num<chair_Max)is
    begin
      num :=num+1;
      put_Line("customer take chair");
    end take_chair;

    function howmany return integer is
    begin
      return num;
    end howmany;

    procedure leave is
    begin
      num :=num-1;
      put_line("customer left ..number of customer waiting :"&num'img);
    end leave;

  end barbershop;
```

```
------------------------------------ Barber Task Type ------------------------------------
task type barber(id:integer) is
  entry haircut;
end barber;
task body barber is
begin
  loop
    if(barbershop.howmany=0)then
      put_line("barber"&id'img&" : I'm going to sleep");
    end if;
    accept haircut do
        put_line("barber"&id'img&" is cutting hair");
        delay 2.0;
      end haircut;
  end loop;
end barber;

---------------------------------- Barbers Instantiation --------------------------------
barber1:barber(1);
barber2:barber(2);
barber3:barber(3);

---------------------------------- Customer Task Type ---------------------------------
task type customer (id:integer);
task body customer is
begin
  delay 1.0;
  put_Line("customer"&id'img&"  arrive to barber shop");
  select
    barbershop.take_chair;
    put_Line("customer"&id'img&" enter");
    select
      barber1.haircut;
    else
      select
        barber2.haircut;
      else
        barber3.haircut;
      end select;
    end select;
    barbershop.leave;
  or
```

```
    delay 3.0;
    put_Line("I wait for 3 second...!! customer"&id'img&" will leave");
  end select;

 end customer;
  ---------------------------------------- Customers Instantiation -------------------------
 customer1:customer(1);
 customer2:customer(2);
 customer3:customer(3);
 customer4:customer(4);
 customer5:customer(5);
 customer6:customer(6);
 customer7:customer(7);
 customer8:customer(8);
 customer9:customer(9);
 customer10:customer(10);

begin
 null;
end;
 ------------------------------------------- End Program... M.N ----------------------------
```

ملاحظات:

- **المهمة تبدأ التنفيذ اعتبارا من لحظة خلقها .. و لذلك نجد إجرائية البرنامج الرئيسي فارغة.**
- يمكن أن نضع كذلك فترة انتظار عظمى للحلاق .. مثلا إذا لم يأت أي زبون خلال 10 ثواني ..!!

  فإنه يغلق المحل و يذهب للبيت .. أو بروح مشوار.

```
Select
        accept  haircut  do
                ........
            end haircut;
        or delay 10.0;
end select;
```

M.N MMNajm@Gmail.com

3 barbers , 10 customers , 5 Chairs :التنفيذ

```
C:\Documents and Settings\M.N>gnatmake barber_main.adb -o barber
gnatbind -x barber_main.ali
gnatlink barber_main.ali -o barber.exe

C:\Documents and Settings\M.N>barber

barber 1 : I'm going to sleep
barber 2 : I'm going to sleep
barber 3 : I'm going to sleep
customer 4  arrive to barber shop
customer take chair
customer 4 enter
barber 1 is cutting hair
customer 5  arrive to barber shop
customer take chair
customer 5 enter
barber 2 is cutting hair
customer 6  arrive to barber shop
customer take chair
customer 6 enter
barber 3 is cutting hair
customer 7  arrive to barber shop
customer take chair
customer 7 enter
customer 1  arrive to barber shop
customer take chair
customer 1 enter
customer 2  arrive to barber shop
customer 3  arrive to barber shop
customer 10  arrive to barber shop
customer 8  arrive to barber shop
customer 9  arrive to barber shop
customer left ..number of customer waiting : 4
customer take chair
customer 2 enter
customer left ..number of customer waiting : 4
customer take chair
customer 3 enter
barber 3 is cutting hair
barber 1 is cutting hair
customer left ..number of customer waiting : 4
customer take chair
customer 10 enter
chairs are full ..I wait for 3 second .. customer 8 will leave
chairs are full ..I wait for 3 second .. customer 9 will leave
```

```
customer left ..number of customer waiting : 4
customer left ..number of customer waiting : 3
barber 3 is cutting hair
customer left ..number of customer waiting : 2
barber 3 is cutting hair
customer left ..number of customer waiting : 1
barber 3 is cutting hair
customer left ..number of customer waiting : 0
barber 3 : I'm going to sleep
```

تعليق على التنفيذ:

- **في البداية .. جميع الحلاقين يغطون في نوم عميق (لا يوجد زبائن ..!!).**
- **يصل أول خمسة زبائن 4,5,6,7,1 : يدخلون مباشرة ،و يجلسون على كراسي و يوقظون الحلاقين و يبدأ الحلاقون بالحلاقة لهم.**
- **نلاحظ بعدها يصل 2,3,10,8,9 ينتظرون على الباب لعدم وجود كراسي فارغة.**
- **ينتهي أحد الزبائن من الحلاقة و يغادر فيدخل 2 .**
- **ثم ينتهي زبون آخر من الحلاقة فيدخل 3.**
- **ثم ينتهي زبون آخر من الحلاقة فيدخل 10.**
- **لاحظ تغير عدد الزبائن المنتظرين و دخول أحد المنتظرين على الباب عند مغادرة كل زبون.**
- **8 و 9 ينتظرون فترة 3 ثواني على الباب و لا يستطيعون الدخول فيغادرون دون حلاقة.**
- **أخيرا ينتهي جميع الزبائن من الحلاقة و يغادرون،و يعود الحلاقون إلى نومهم و يستغرقون في أحلامهم**

ملاحظة :نلاحظ أنه كلما زدنا مدة انتظار الزبائن على الباب يقل عدد الزبائن الذين يغادرون دون حلاقة ،و يظهر ذلك جليا من خلال تغيير المدة و التنفيذ.

M.N Moustafa-MN@hotmail.com

الحل باستخدام **Requeue** :

```
-- Barber program
--this solution for Barberer problem by Ada with PU & Requeue
--Author Moustafa Najm ........... M.N : Moustafa-MN@hotmail.com

with Ada.Text_IO;
use Ada.Text_IO;

procedure main is

  ---------------------- Protected Unit Shop ---------------------
  protected shop is
    entry enter;
    procedure leave;

  private
    count:integer:=0;
    barber:integer:=3;
    Max:integer:=5;
    entry share;
  end shop;

  protected body shop is

    entry enter when count<Max is
    begin
      count:=count+1;
      put_line("custumer enter .. and wait now to shar");
      requeue share;
    end enter;

    entry share when barber>0 is
    begin
      barber:= barber-1;-- get haircut
    end share;

    procedure leave is
    begin
      count :=count-1;
      barber:=barber+1;
    end leave;

  end shop;
```

```
 ----------------------Barber Task --------------------------------
  task  Barber; -- there is no thing to do .. we can remove
  task body barber is
  begin
    null;
  end;
  ---------------------- Customer Task Type ------------------------
  task Type customer(id:integer);
  task body customer is
  begin
    select
      shop.enter;
      put_Line("customer"&id'img&" enter .. and he is share now");
      delay 2.0;
      shop.leave;
      put_Line("customer"&id'img&" shared and left");
    or
      delay 3.0;
      put_Line("I wait for 3 second...!! customer"&id'img&" will leave");
    end select;
  end customer;
  ------------------------ Customers Instantiation -----------------
  customer1:customer(1);
  customer2:customer(2);
  customer3:customer(3);
  customer4:customer(4);
  customer5:customer(5);
  customer6:customer(6);
  customer7:customer(7);
  customer8:customer(8);
  customer9:customer(9);
  customer10:customer(10);

begin
  null;
end main;
  ----------------------------- End Main ... M.N -------------------
```

التنفيذ :

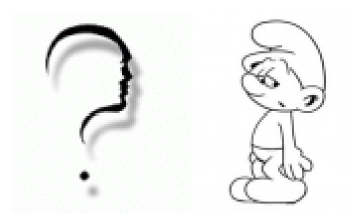
```
custumer enter .. and wait now to shar
customer 1 enter .. and he is share now
custumer enter .. and wait now to shar
customer 2 enter .. and he is share now
custumer enter .. and wait now to shar
customer 3 enter .. and he is share now
```

```
custumer enter .. and wait now to shar
custumer enter .. and wait now to shar
customer left ..number of customer waiting : 4
custumer enter .. and wait now to shar
customer 4 enter .. and he is share now
customer 1 shared and left
customer left ..number of customer waiting : 4
custumer enter .. and wait now to shar
customer 5 enter .. and he is share now
customer 2 shared and left
customer left ..number of customer waiting : 4
custumer enter .. and wait now to shar
customer 6 enter .. and he is share now
customer 3 shared and left
I wait for 3 second...!! customer 9 will leave
I wait for 3 second...!! customer 10 will leave
customer left ..number of customer waiting : 4
customer 7 enter .. and he is share now
customer 4 shared and left
customer left ..number of customer waiting : 3
customer 8 enter .. and he is share now
customer 6 shared and left
customer left ..number of customer waiting : 2
customer 5 shared and left
customer left ..number of customer waiting : 1
customer 8 shared and left
customer left ..number of customer waiting : 0
customer 7 shared and left
```

ملاحظة :

لاحظ هنا أن الحلاق Barber ليس له أي أهمية ،حتى أنه يمكن حذف المهمة Barber بشكل كامل.

(حيث يهمنا فقط عدد الحلاقين ،و هو عبارة عن متحول موجود ضمن الوحدة المحمية).



لا تحاول أن تجعل ملابسك أغلى شيء فيك ، حتّى لا تجد

نفسك يوماً أرخص مما ترتديه

الحل باستخدام Semaphore

```
-- Barbershop program Using Ada (Semaphore)
--Author Moustafa Najm ... M.N : Moustafa-MN@hotmail.com

with Ada.Text_IO;
use Ada.Text_IO;

procedure barbershop is

   protected type semaphore is

 procedure decrement ;
 procedure up;
 entry down;
 private value: integer:=1;
   end semaphore;

   protected body semaphore is
      procedure decrement is
      begin
         value:= value+1;
      end decrement;
      procedure up is
      begin
         value:=value+1;
      end up;
      entry down when value>0 is
      begin
         value:=value-1;
      end down;
   end semaphore;

   mutex , barber , customer : semaphore;
   customers: integer:=0;
   max :constant integer:=5;

   task type barberTask;

   task body barberTask is
   begin
      customer.decrement;
      barber.decrement;
      loop
         customer.down;
         delay 2.0;
         barber.up;
      end loop;
   end barberTask;

   task type customerTask (id:integer);
   task body customerTask is
   begin
      --we use loop to round customers and generate infinite customers
      loop
         mutex.down;
      if customers = max then
```

```
         mutex.up;
         put_line("waiting room is full customer"&id'img&" will leave");

      else
         customers:=customers+1;
         mutex.up;
         customer.up;
         barber.down;
         mutex.down;
         customers:=customers-1;
         mutex.up;
         put_line("Customer "& id'img &" is getting a hair-cut");
      end if;
         delay 2.0;
      end loop;
   end customerTask;

   bar1: barberTask;
   bar2: barberTask;
   bar3: barberTask;

   customer1:customerTask(1);
   customer2:customerTask(2);
   customer3:customerTask(3);
   customer4:customerTask(4);
   customer5:customerTask(5);
   customer6:customerTask(6);
   customer7:customerTask(7);
   customer8:customerTask(8);
   customer9:customerTask(9);
   customer10:customerTask(10);

begin
   null;
end barbershop;
```


التنفيذ:

```
Customer  1 is getting a hair-cut
Customer  2 is getting a hair-cut
Customer  3 is getting a hair-cut
Customer  4 is getting a hair-cut
waiting room is full customer 10 will leave
Customer  5 is getting a hair-cut
Customer  6 is getting a hair-cut
Customer  7 is getting a hair-cut
waiting room is full customer 2 will leave
waiting room is full customer 10 will leave
waiting room is full customer 5 will leave
waiting room is full customer 6 will leave
waiting room is full customer 7 will leave
Customer  8 is getting a hair-cut
Customer  9 is getting a hair-cut
Customer  3 is getting a hair-cut
Customer  4 is getting a hair-cut
```

الحل باستخدام Package

```
-- Barbershop program Using Ada (Package)
-- Author Moustafa Najm ........... M.N : Moustafa-MN@hotmail.com

-- This file contains the interface definition for the Barbershop
-- Barbershop.Ads
generic
   Num_Chairs : Positive;

package BarberShop is

   subtype Customer is Positive;
   type Barber_Queue is array(Positive range 1..Num_Chairs)
                    of Customer;

   protected Barber_Chairs is

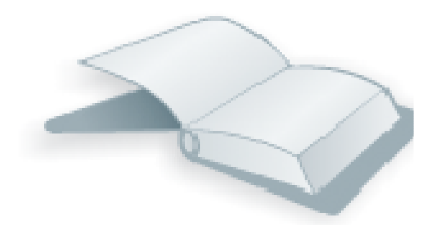
      function Is_Full return Boolean;
      entry Take_Chair(The_Customer : in Customer);
      entry Wait_For_Customer(The_Customer : out Customer);

   private
      Queue : Barber_Queue;
      Next_Available : Positive := 1;
      Next_Customer : Positive := 1;
      Num_Waiting : Natural := 0;
   end Barber_Chairs;

   task Barber is
      entry Stop;
   end Barber;

   task Demo_Master;

end BarberShop;
```



```
-- This file contains the implementation of the Barbershop
-- BarberShop.adb

with Ada.Text_Io;
use Ada.Text_Io;
   ---------------------- Package Body BarberShop ---------------------------
package body BarberShop is

   --------------------- Protected Unit barber_chairs ---------------------
   protected body Barber_Chairs is

      --------------------function Is_Full ------------------------
      function Is_Full return Boolean is
      begin
         return Num_Waiting = Num_Chairs;
      end Is_Full;

      --------------------Entry Take_chair ------------------------
      entry Take_Chair (The_Customer : in Customer ) when not Is_Full is
      begin
         Queue(Next_Available) := The_Customer;
         if Next_Available = Num_Chairs then
            Next_Available := 1;
         else
            Next_Available := Next_Available + 1;
         end if;
         Num_Waiting := Num_Waiting + 1;
      end Take_Chair;

      --------------------Entry Wait_For_Costomer ---------------------
      entry Wait_For_Customer ( The_Customer : out Customer ) when Num_Waiting >
0 is
      begin
         The_Customer := Queue(Next_Customer);
         if Next_Customer = Num_Chairs then
            Next_Customer := 1;
         else
            Next_Customer := Next_Customer + 1;
         end if;

      end Wait_For_Customer;

   end Barber_Chairs;

      ------------------------- Task Type Barber ---------------------------
   task body Barber is
      Current_Customer : Customer;

   begin
      loop
```

```
         select
            Barber_Chairs.Wait_For_Customer(Current_Customer);
            Put_Line("Serving customer" & Customer'Image(Current_Customer));
            delay 2.0; -- a 2 second hair cut
         else
            delay 1.0; -- sleep one second waiting for a customer
            select
               accept Stop;
               exit;
            else
               delay 0.0;
                  Put_Line("ZZZZZZZZZZ");
            end select;
         end select;
      end loop;

   end Barber;

      ------------------------- Task Demo_Master ----------------------------
   task body Demo_Master is
      Cust_Max : constant Customer := 20;

   begin
      delay 2.0;
      for Cust in 1..Cust_Max loop
         delay 0.5;
         if not Barber_Chairs.Is_Full then
            Barber_Chairs.Take_Chair(Cust);
            Put_Line("Customer" & Customer'Image(Cust) &" took a chair
                     in the barber shop.");
         else
            Put_Line("Shop full, customer" & Customer'Image(Cust) &"left shop");
         end if;
      end loop;
      delay 3.0;
      Barber.Stop;
   end Demo_Master;
end BarberShop;

   ----------------------End Package Body BarberShop ----------------------
-- This file contains Main Procedure
-- Barber_Main.adb

with Barbershop;
procedure Barber_Main is
   package Small_Shop is new BarberShop(Num_Chairs => 5);

begin
   null;
end Barber_Main;
```

التنفيذ :

ZZZZZZZZZZ
ZZZZZZZZZZ
Customer 1 took a chair in the barber shop.
Customer 2 took a chair in the barber shop.
ZZZZZZZZZZ
Serving customer 1
Customer 3 took a chair in the barber shop.
Customer 4 took a chair in the barber shop.
Customer 5 took a chair in the barber shop.
Serving customer 2
Customer 6 took a chair in the barber shop.
Customer 7 took a chair in the barber shop.
Shop full, customer 8 left shop
Shop full, customer 9 left shop
Serving customer 3
Customer 10 took a chair in the barber shop.
Shop full, customer 11 left shop
Shop full, customer 12 left shop
Shop full, customer 13 left shop
Serving customer 4
Customer 14 took a chair in the barber shop.
Shop full, customer 15 left shop
Shop full, customer 16 left shop
Shop full, customer 17 left shop
Serving customer 5
Customer 18 took a chair in the barber shop.
Shop full, customer 19 left shop
Shop full, customer 20 left shop
Serving customer 6
Serving customer 7
Serving customer 10
Serving customer 14
Serving customer 18



تم بعونه تعالى ....

Moustafa Najm **Moustafa-MN@hotmail.com**